

From Analog to Discrete Signals

As you may recall, the goal of this course was to introduce some of the tools of applied mathematics with an underlying theme of finding the connection between analog and discrete signals. We began our studies with Fourier series, which provided representations of periodic functions. We then moved on to the study of Fourier transforms, which can represent functions defined over all space. Such functions can be used to describe analog signals. However, we cannot record and store analog signals. There is an uncountable amount of information in analog signals. We have to record the signal for a finite amount of time and even then we can only store samples of the signal over that time interval. The resulting signal is discrete. These discrete signals are represented using the Discrete Fourier Transform (DFT). In this chapter we will discuss the general steps of relating analog, periodic and discrete signals. Then we will go further into the properties of the Discrete Fourier Transform (DFT) and in Section 9 we will apply what we have learned to the study of real signals.

8.1 Analog to Periodic Signals

For now we consider analog signals. As an example, we consider an analog signal and its Fourier transform as shown in Figure 8.1. We see that the analog signal can be described as a piecewise continuous function defined over an infinite time interval. The resulting Fourier transform is also piecewise continuous and defined over an infinite interval of frequencies. We can represent the signal, $f(t)$, and its transform, $\hat{f}(\omega)$, using the Fourier transform and the inverse Fourier transform, respectively:

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(\omega) e^{-i\omega t} d\omega, \quad (8.1)$$

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(t) e^{i\omega t} dt. \quad (8.2)$$

Note that the figures in this section are drawn as if the transform is real. However, in general they are not and we will investigate how this can be handled in the last chapter.

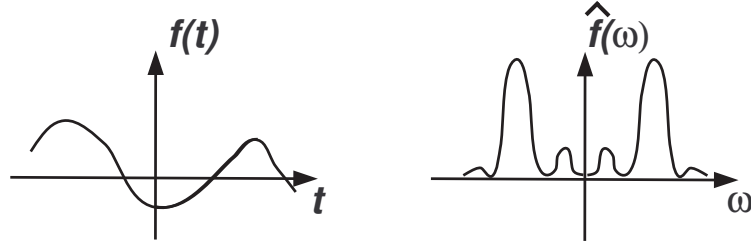


Fig. 8.1. Plot of an analog signal $f(t)$ and its Fourier transform $\hat{f}(\omega)$.

Real signals cannot be studied on an infinite interval. Such signals are captured as data over a finite time interval. Let's assume that the recording starts at $t = 0$. Then the interval can be written as $[0, T]$, where T will be called the *record length*.

The natural representation of a function, $f(t)$ for $t \in [0, T]$ is obtained by extending the signal to a periodic signal, knowing that the physical signal is only defined on $[0, T]$. Recall that periodic functions can be modeled by a Fourier series:

$$f_p(t) = \sum_{n=-\infty}^{\infty} c_n e^{-i\omega_n t}, \quad (8.3)$$

$$c_n = \frac{1}{T} \int_0^T f_p(t) e^{i\omega_n t} dt. \quad (8.4)$$

Here we have defined the discrete angular frequency $\omega_n = \frac{2\pi n}{T}$. The associated frequency is then $\nu_n = \frac{n}{T}$.

Given that $f_p(t)$ is a periodic function, we would like to relate its Fourier series representation to the Fourier transform $\hat{f}_p(\omega)$ of the corresponding analog signal $f(t)$. This is done by simply computing the Fourier transform of $f_p(t)$. Thus,

$$\begin{aligned} \hat{f}_p(\omega) &= \int_{-\infty}^{\infty} f_p(t) e^{i\omega t} dt \\ &= \int_{-\infty}^{\infty} \left(\sum_{n=-\infty}^{\infty} c_n e^{-i\omega_n t} \right) e^{i\omega t} dt \\ &= \sum_{n=-\infty}^{\infty} c_n \int_{-\infty}^{\infty} e^{i(\omega - \omega_n)t} dt. \end{aligned} \quad (8.5)$$

Recalling

$$\int_{-\infty}^{\infty} e^{i\omega x} dx = 2\pi\delta(\omega),$$

we have that

$$\hat{f}_p(\omega) = \sum_{n=-\infty}^{\infty} 2\pi c_n \delta(\omega - \omega_n). \quad (8.6)$$

Thus, the Fourier transform of a periodic function is a series of spikes at discrete frequencies $\omega_n = \frac{2\pi n}{T}$ of strength $2\pi c_n$. This is represented in Figure 8.2.

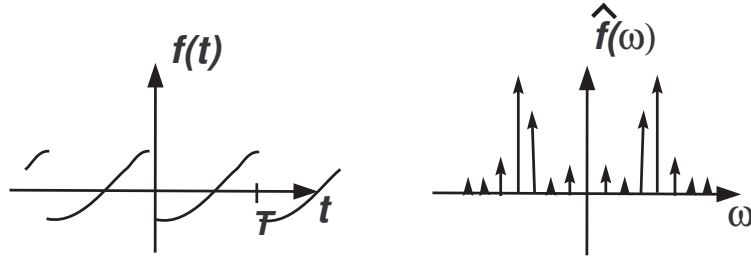


Fig. 8.2. A periodic signal contains a discrete number of frequencies.

We now determine a relationship between the Fourier transform of $f_p(t)$ and the Fourier coefficients. Namely, we have

$$\hat{f}_p(\omega_k) = \sum_{n=-\infty}^{\infty} 2\pi c_n \delta(\omega_k - \omega_n) = 2\pi c_k.$$

Thus, we can write

$$\begin{aligned} \hat{f}_p(\omega) &= \sum_{n=-\infty}^{\infty} \hat{f}_p(\omega_n) \delta(\omega - \omega_n) \\ &= \sum_{n=-\infty}^{\infty} \hat{f}_p(\omega) \delta(\omega - \omega_n) \\ &= \hat{f}_p(\omega) \sum_{n=-\infty}^{\infty} \delta\left(\omega - \frac{2\pi n}{T}\right). \end{aligned} \quad (8.7)$$

This shows that the Fourier transform of a periodic signal is represented by a set of spikes at frequencies $\omega_n = \frac{2\pi n}{T}$ as was shown in Figure 8.2. In Figure 8.3 the discrete spectrum is superimposed on the continuous spectrum emphasizing this connection.

The sum in the last line of Equation (8.7) is a special function, the Dirac comb function, which we discuss further in the next section.

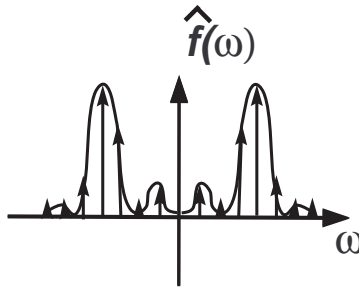


Fig. 8.3. The discrete spectrum obtained from the Fourier transform of the periodic extension of $f(t)$, $t \in [0, T]$ is superimposed on the continuous spectrum of the analog signal.

8.2 The Comb Function

A function that often occurs in signal analysis is the *comb function* defined by

$$\text{comb}_a(t) = \sum_{n=-\infty}^{\infty} \delta(t - na). \quad (8.8)$$

This function is simply a set of translated delta function spikes as shown in Figure 8.4. It is also called an *impulse train* or a *sampling function*. It is a periodic distribution and care needs to be taken in studying the properties of this function. We will show how the comb function can be used to relate analog and finite length signals.

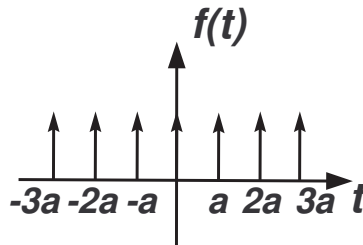


Fig. 8.4. The Dirac comb function, $\text{comb}_a(t) = \sum_{n=-\infty}^{\infty} \delta(t - na)$.

In some fields, the comb function is written using the Cyrillic uppercase *Sha* function,

$$\text{Ш}(x) = \sum_{n=-\infty}^{\infty} \delta(x - n).$$

We note a few properties of the Sha function:

$$\begin{aligned}\text{III}(ax) &= \frac{1}{|a|} \sum_{n=-\infty}^{\infty} \delta\left(x - \frac{n}{a}\right) \\ &= \frac{1}{|a|} \text{comb}_{\frac{1}{a}}(t).\end{aligned}\tag{8.9}$$

$$\text{III}(x+n) = \text{III}(ax).\tag{8.10}$$

Also, we have the *Sampling Property*,

$$\text{III}(x)f(x) = \sum_{n=-\infty}^{\infty} f(n)\delta(x-n)\tag{8.11}$$

and the *Replicating Property*

$$(\text{III} * f)(x) = \sum_{n=-\infty}^{\infty} f(x-n).\tag{8.12}$$

The comb function inherits these properties, since it can be written using the Sha function as

$$\text{comb}_T(t) = \sum_{n=-\infty}^{\infty} \delta(t-nT) = \frac{1}{|T|} \text{III}\left(\frac{t}{T}\right).$$

In the following we will only use the comb function.

We note that $\text{comb}_T(t)$ has period T :

$$\begin{aligned}\text{comb}_T(t+T) &= \sum_{n=-\infty}^{\infty} \delta(t+T-nT) \\ &= \sum_{n=-\infty}^{\infty} \delta(t-nT) = \text{comb}_T(t).\end{aligned}\tag{8.13}$$

Thus, the comb function has a Fourier series representation,

$$\text{comb}_T(t) = \sum_{n=-\infty}^{\infty} c_n e^{-2\pi i n t / T},$$

where

$$c_n = \frac{1}{T} \int_{-T}^T \delta(t-nT) e^{-2\pi i n t / T} dt = \frac{1}{T}.$$

Therefore, we have the Fourier representation

$$\text{comb}_T(t) = \frac{1}{T} \sum_{n=-\infty}^{\infty} e^{-2\pi i n t / T}.$$

Since $\text{comb}_T(t)$ has period T , the convolution of a function with $\text{comb}_T(x)$ is a periodic function with period T . This is easily seen by

$$\begin{aligned} (f * \text{comb}_T)(t + T) &= \int_{-\infty}^{\infty} f(\tau) \text{comb}_T(t + T - \tau) d\tau \\ &= \int_{-\infty}^{\infty} f(\tau) \text{comb}_T(t - \tau) d\tau \\ &= (f * \text{comb}_T)(t). \end{aligned} \quad (8.14)$$

Therefore, we have

$$f_p(t) = (f * \text{comb}_T)(t).$$

We need only show that the Fourier transform of f_p can be written as a series of delta function spikes. In order to show this we have to compute the Fourier transform of this convolution, which is a product of transforms of f and comb_T . We show this by computing the Fourier transform directly:

$$\begin{aligned} F[\text{comb}_a(t)] &= \int_{-\infty}^{\infty} \text{comb}_a(t) e^{i\omega t} dt \\ &= \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(t - na) e^{i\omega t} dt \\ &= \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} \delta(t - na) e^{i\omega t} dt \\ &= \sum_{n=-\infty}^{\infty} e^{i\omega na} \\ &= \sum_{n=-\infty}^{\infty} e^{2\pi i n \omega / (2\pi/a)} \\ &= \frac{2\pi}{a} \text{comb}_{\frac{2\pi}{a}}(\omega). \end{aligned} \quad (8.15)$$

The Fourier transform of a function of period T , $f_p(t)$, is then computed as

$$\begin{aligned} \hat{f}_p(\omega) &= F[f * \text{comb}_T] \\ &= \hat{f}(\omega) F[\text{comb}_T](\omega) \\ &= \frac{2\pi}{T} \hat{f}(\omega) \text{comb}_{\frac{2\pi}{T}}(\omega). \end{aligned} \quad (8.16)$$

Thus, the spectrum is a series of scaled delta functions.

Next we carry out a direct computation of the convolution integral. We do this by first considering the convolution of a function $f(t)$ with a shifted Dirac delta function, $\delta_a(t) = \delta(t - a)$. This convolution is easily computed as

$$(f * \delta_a)(t) = \int_{-\infty}^{\infty} f(t - \tau) \delta(\tau - a) d\tau = f(t - a).$$

Therefore, we have found that the convolution of a function with one shifted delta function is a copy of $f(t)$ that is shifted by a .

Now convolve $f(t)$ with the comb function:

$$\begin{aligned} (f * \text{comb}_a)(t) &= \int_{-\infty}^{\infty} f(t - \tau) \text{comb}_a(\tau) d\tau \\ &= \int_{-\infty}^{\infty} f(t - \tau) \sum_{n=-\infty}^{\infty} \delta(\tau - na) d\tau \\ &= \sum_{n=-\infty}^{\infty} (f * \delta_{na})(t) \\ &= \sum_{n=-\infty}^{\infty} f(t - na). \end{aligned} \tag{8.17}$$

The convolution of a function $f(t)$ with a comb function is then the sum of shifted copies of $f(t)$, as shown in Figure 8.5. If the function has compact support on $[-\frac{a}{2}, \frac{a}{2}]$, i.e., the function is zero for $|t| > 1/a$, then the convolution with the comb function will be periodic.

Returning to the Fourier transform of our periodic function, we found that the transform produced a series of spikes. This series of spikes is the transform of the convolution of $f(t)$ and a comb function. This is essentially a sampling in frequency space. A similar result is obtained if we instead had a periodic Fourier transform due to a sampling in t -space. Combining both discrete functions in t and ω spaces, we have discrete signals, as will be described next.

8.3 Discrete Signals

Finally, we would like to sample our signal at a discrete set of equally spaced times, $t_n \in [0, T]$. This is how one normally records signals. One samples the signal with a sampling frequency ν_s , such as ten samples per second. Therefore, the values are recorded in time steps of $T_s = 1/\nu_s$. We can model this sampling at discrete time points by multiplying $f(t)$ by the comb function $\text{comb}_{T_s}(t)$. The Fourier transform will yield a convolution of the Fourier transform of $f(t)$ with the Fourier transform of the comb function. But this is a convolution of $\hat{f}(\omega)$ with another comb function, since the transform of a comb function is a comb function. Therefore, we will obtain a periodic representation in Fourier space.

In collecting data, we not only sample at a discrete set of points, but we also sample for a finite length of time. By sampling like this, we will not gather

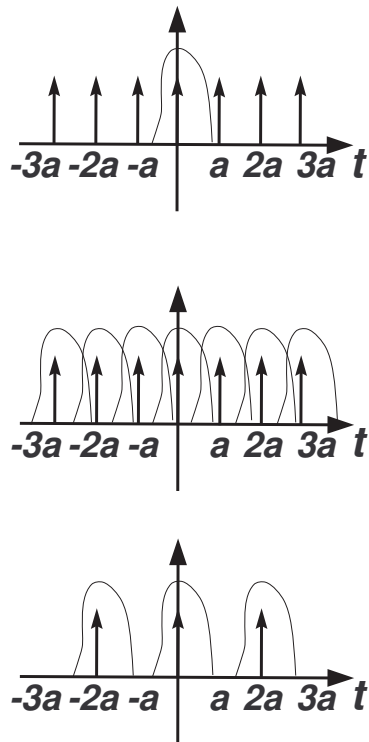


Fig. 8.5. The convolution of $f(t)$ with the comb function, $\text{comb}_a(t)$. The first plot shows the function and the comb function. In the second of these plots we add the sum of several translations of $f(t)$. Incorrect sampling will lead to overlap in the translates and cause problems like aliasing. In the last of these plots, one has no overlap and the periodicity is evident.

enough information to obtain the high frequencies in a signal. Thus, there will be a natural cutoff in the spectrum of the signal. This is represented in Figure 8.6. This process will lead to the Discrete Fourier Transform, the topic in the next chapter.

Once again, we can use the comb function in the analysis of this process. We define a discrete signal as one which is represented by the product of a periodic function sampled at discrete times. This suggests the representation

$$f_d(t) = f_p(t)\text{comb}_{T_s}(t). \quad (8.18)$$

Here T_s denotes the period of the sampling and $f_p(t)$ has the form

$$f_p(t) = \sum_{n=-\infty}^{\infty} c_n e^{-i\omega_n t}.$$

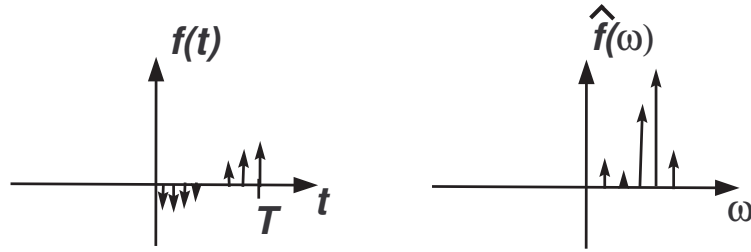


Fig. 8.6. Sampling the original signal at a discrete set of times defined on a finite interval leads to a discrete set of frequencies in the transform that are restricted to a finite interval of frequencies.

The Fourier transform of $f_d(t)$ can be computed:

$$\begin{aligned}
 \hat{f}_d(\omega) &= \int_{-\infty}^{\infty} f_p(t) \text{comb}_{T_s}(t) e^{i\omega t} dt \\
 &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} \hat{f}_p(\mu) e^{-i\mu t} d\mu \right] \text{comb}_{T_s}(t) e^{i\omega t} dt \\
 &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}_p(\mu) \underbrace{\int_{-\infty}^{\infty} \text{comb}_{T_s}(t) e^{i(\omega-\mu)t} dt}_{\text{Transform of } \text{comb}_{T_s} \text{ at } \omega-\mu} d\mu \\
 &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}_p(\mu) \frac{2\pi}{T_s} \text{comb}_{\frac{2\pi}{T_s}}(\omega - \mu) d\mu \\
 &= \frac{1}{T_s} \sum_{n=-\infty}^{\infty} \hat{f}_p\left(\omega - \frac{2\pi}{T_s} n\right). \tag{8.19}
 \end{aligned}$$

We note that in the middle of this computation we find a convolution integral:

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}_p(\mu) \frac{2\pi}{T_s} \text{comb}_{\frac{2\pi}{T_s}}(\omega - \mu) d\mu = \left(\hat{f}_p * \text{comb}_{\frac{2\pi}{T_s}} \right) (\omega)$$

Also, it is easily seen that $\hat{f}_d(\omega)$ is a periodic function with period $\frac{2\pi}{T_s}$:

$$\hat{f}_d\left(\omega + \frac{2\pi}{T_s}\right) = \hat{f}_d(\omega).$$

We have shown that sampling a function $f_p(t)$ with sampling frequency $\nu_s = 1/T_s$, one obtains a periodic spectrum, $\hat{f}_d(\omega)$.

8.3.1 Summary

In this chapter we have taken an analog signal defined for $t \in (-\infty, \infty)$ and shown that restricting it to interval $t \in [0, T]$ leads to a periodic function of

period T ,

$$f_p(t) = (f * \text{comb}_T)(t),$$

whose spectrum becomes discrete:

$$\hat{f}_p(\omega) = \hat{f}_p(\omega) \sum_{n=-\infty}^{\infty} \delta\left(\omega - \frac{2\pi n}{T}\right). \quad (8.20)$$

We then sampled this function at discrete times,

$$f_d(t) = f_p(t) \text{comb}_{T_s}(t).$$

The Fourier transform of this discrete signal was found as

$$\hat{f}_d(\omega) = \frac{1}{T_s} \sum_{n=-\infty}^{\infty} \hat{f}_p\left(\omega - \frac{2\pi n}{T_s}\right). \quad (8.21)$$

This function is periodic with period $\frac{2\pi}{T_s}$.

In Figure 8.7 we summarize the steps for going from analog signals to discrete signals. In the next chapter we will investigate the Discrete Fourier Transform.

8.4 The Discrete Fourier Transform

Often one is interested in determining the frequency content of signals. Signals are typically represented as time dependent functions. Real signals are continuous, or analog signals. However, through sampling the signal by gathering data, the signal does not contain high frequencies and is finite in duration. The data is then discrete and the corresponding frequencies are discrete and bounded. Thus, in the process of gathering data, one may seriously affect the frequency content of the signal. This is true for a simple superposition of signals with fixed frequencies. The situation becomes more complicated if the data has an overall non-constant (in time) trend or even exists in the presence of noise.

As described earlier in this chapter, we have seen that by restricting our data to a time interval $[0, T]$ for record length T , and periodically extending the data for $t \in (-\infty, \infty)$, one generates a periodic function of infinite duration at the cost of losing data outside the fundamental period. This is not unphysical, as data is typically taken over a finite time period. In addition, if one samples a finite number of values of the data on this finite interval, then the signal will contain frequencies limited to a finite range of values.

This process, as reviewed in the last section, leads us to a study of what is called the *Discrete Fourier Transform*, or *DFT*. We will investigate the discrete Fourier transform (DFT) in both trigonometric and exponential form.

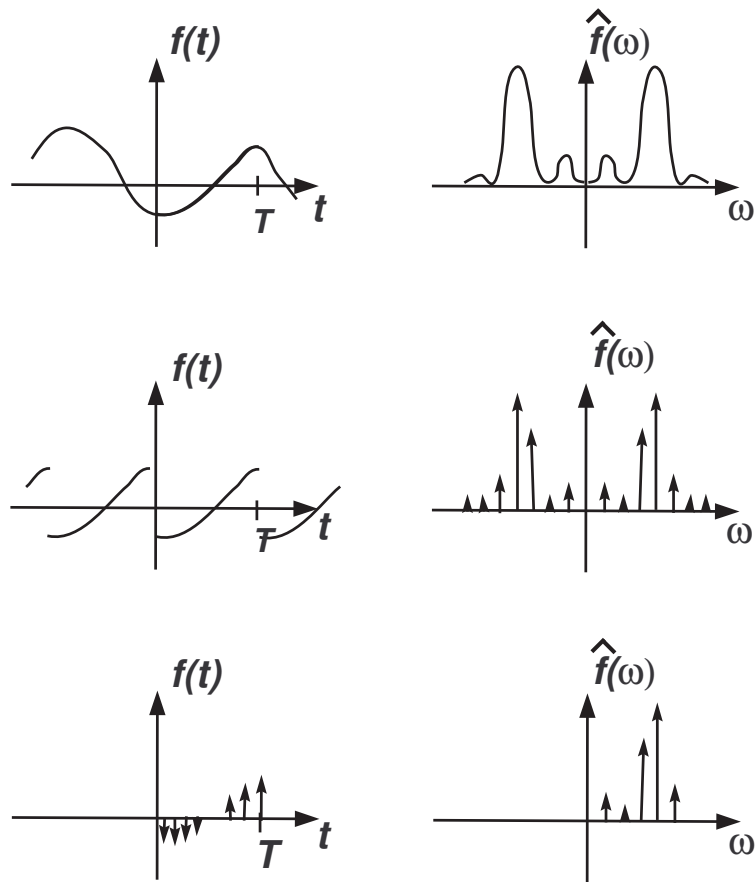


Fig. 8.7. Summary of transforming analog to discrete signals. One starts with a continuous signal $f(t)$ defined on $(-\infty, \infty)$ and a continuous spectrum. By only recording the signal over a finite interval $[0, T]$, the recorded signal can be represented by its periodic extension. This in turn forces a discretization of the transform as shown in the second row of figures. Finally, by restricting the range of the sampled, as shown in the last row, the original signal appears as a discrete signal. This is also interpreted as the sampling of an analog signal leads to a restricted set of frequencies in the transform.

Recall that in using Fourier series one seeks a representation of the signal, $y(t)$, valid for $t \in [0, T]$, as

$$y(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} [a_n \cos \omega_n t + b_n \sin \omega_n t], \quad (8.22)$$

where the angular frequency is given by $\omega_n = 2\pi f_n = \frac{2\pi n}{T}$. Note: In discussing signals, we will use $y(t)$ instead of $f(t)$, allowing us to use f to denote the frequency (in Hertz) without confusion.

The frequency content of a signal for a particular f_n is contained in both the cosine and sine terms when $a_n, b_n \neq 0$. So, one may desire to combine these terms. This is easily done by using trigonometric identities. Namely, we show that

$$a_n \cos \omega_n t + b_n \sin \omega_n t = C_n \cos(\omega_n t + \phi), \quad (8.23)$$

where ϕ is called the *phase shift*. Recalling that

$$\cos(\omega_n t + \phi) = \cos \omega_n t \cos \phi - \sin \omega_n t \sin \phi, \quad (8.24)$$

one has

$$a_n \cos \omega_n t + b_n \sin \omega_n t = C_n \cos \omega_n t \cos \phi - C_n \sin \omega_n t \sin \phi.$$

Equating the coefficients of $\sin \omega_n t$ and $\cos \omega_n t$ in this expression, we obtain

$$a_n = C_n \cos \phi, \quad b_n = -C_n \sin \phi.$$

Therefore,

$$C_n = \sqrt{a_n^2 + b_n^2} \quad \text{and} \quad \tan \phi = -\frac{b_n}{a_n}.$$

Recall that we had used orthogonality arguments in order to determine the Fourier Coefficients ($a_n, n = 0, 1, 2, \dots$ and $b_n, n = 1, 2, 3, \dots$). In particular, using the orthogonality of the trigonometric basis, we found that

$$\begin{aligned} a_n &= \frac{2}{T} \int_0^T y(t) \cos \omega_n t \, dt, \quad n = 0, 1, 2, \dots \\ b_n &= \frac{2}{T} \int_0^T y(t) \sin \omega_n t \, dt, \quad n = 1, 2, \dots \end{aligned} \quad (8.25)$$

In the next section we will introduce the trigonometric version of the Discrete Fourier Transform. Its appearance is similar to that of the above Fourier series representation. However, we will need to do a bit of work to obtain the discrete Fourier coefficients using discrete orthogonality.

8.4.1 Discrete Series

For the Fourier series analysis of a signal, we had restricted time to the interval $[0, T]$, leading to a Fourier series with discrete frequencies and a periodic function of time. In reality, taking data can only be done at certain frequencies, thus eliminating high frequencies. Such a restriction on the frequency leads to a discretization of the data in time. Another way to view this is that when recording data we sample at a finite number of time steps, limiting our ability to collect data with large oscillations. Thus, we not only have discrete frequencies but we also have discrete times.

We first note that the data is sampled at N equally spaced times

$$t_n = n\Delta t, \quad n = 0, 1, \dots, N-1,$$

where Δt is the time increment. For a record length of T , we have $\Delta t = T/N$. We will denote the data at these times as $y_n = y(t_n)$.

The DFT representation that we are seeking takes the form:

$$y_n = \frac{1}{2}a_0 + \sum_{p=1}^M [a_p \cos \omega_p t_n + b_p \sin \omega_p t_n], \quad n = 0, 1, \dots, N-1. \quad (8.26)$$

The trigonometric arguments are given by $\omega_p t_n = \frac{2\pi p n}{N}$. Note that $p = 1, \dots, M$, thus allowing only for frequencies $f_p = \frac{\omega_p}{2\pi} = \frac{p}{T}$. Or, we could write

$$f_p = p\Delta f$$

for

$$\Delta f = \frac{1}{T}.$$

We need to determine M and the unknown coefficients. As for the Fourier series, we will need some orthogonality relations, but this time our the orthogonality statement will consist of a sum and not an integral.

Since there are N sample values, (8.26) gives us a set of N equations for the unknown coefficients. Therefore, we should have N unknowns. For N samples, we want to determine N unknown coefficients $a_0, a_1, \dots, a_{N/2}$ and $b_1, \dots, b_{N/2-1}$. Thus, we need to fix $M = \frac{N}{2}$. Often the coefficients b_0 and $b_{N/2}$ are included for symmetry. Note that the corresponding sine function factors evaluate to zero, leaving these two coefficients arbitrary. Thus, we can take them to be zero when they are included.

We claim that the DFT coefficients are given by

$$\begin{aligned} a_p &= \frac{2}{N} \sum_{n=1}^N y(t_n) \cos\left(\frac{2\pi p n}{N}\right), \quad p = 1, \dots, N/2 - 1 \\ b_p &= \frac{2}{N} \sum_{n=1}^N y(t_n) \sin\left(\frac{2\pi p n}{N}\right), \quad p = 1, 2, \dots, N/2 - 1 \\ a_0 &= \frac{1}{N} \sum_{n=1}^N y(t_n), \\ a_{N/2} &= \frac{1}{N} \sum_{n=1}^N y(t_n) \cos n\pi, \\ b_0 &= b_{N/2} = 0 \end{aligned} \quad (8.27)$$

8.4.2 Discrete Orthogonality

The derivation of the discrete Fourier coefficients can be done using the discrete orthogonality of the discrete trigonometric basis similar to the derivation of the above Fourier coefficients for the Fourier series. We first prove the following

$$\sum_{n=0}^{N-1} \cos\left(\frac{2\pi nk}{N}\right) = \begin{cases} 0, & k = 1, \dots, N-1 \\ N, & k = 0, N \end{cases} \quad (8.28)$$

$$\sum_{n=0}^{N-1} \sin\left(\frac{2\pi nk}{N}\right) = 0, \quad k = 0, \dots, N$$

This can be done more easily using the exponential form,

$$\sum_{n=0}^{N-1} \cos\left(\frac{2\pi nk}{N}\right) + i \sum_{n=0}^{N-1} \sin\left(\frac{2\pi nk}{N}\right) = \sum_{n=0}^{N-1} e^{2\pi ink/N}, \quad (8.29)$$

by using Euler's formula, $e^{i\theta} = \cos\theta + i\sin\theta$ for each term in the sum.

The exponential sum is the sum of a geometric progression. Namely, we note that

$$\sum_{n=0}^{N-1} e^{2\pi ink/N} = \sum_{n=0}^{N-1} \left(e^{2\pi ik/N}\right)^n.$$

Recall from our chapter on sequences and series that a geometric progression is a sum of the form $S_N = \sum_{k=0}^{N-1} ar^k$. This is a sum of N terms in which consecutive terms have a constant ratio, r . The sum is easily computed. One multiplies the sum S_N by r and subtracts the resulting sum from the original sum to obtain

$$S_N - rS_N = (a + ar + \dots + ar^{N-1}) - (ar + \dots + ar^N + ar^N) = a - ar^N. \quad (8.30)$$

Factoring on both sides of this chain of equations yields the desired sum,

$$S_N = \frac{a(1 - r^N)}{1 - r}. \quad (8.31)$$

Thus, we have

$$\begin{aligned} \sum_{n=0}^{N-1} e^{2\pi ink/N} &= \sum_{n=0}^{N-1} \left(e^{2\pi ik/N}\right)^n \\ &= 1 + e^{2\pi ik/N} + \left(e^{2\pi ik/N}\right)^2 + \dots + \left(e^{2\pi ik/N}\right)^{N-1} \\ &= \frac{1 - \left(e^{2\pi ik/N}\right)^N}{1 - e^{2\pi ik/N}} \\ &= \frac{1 - e^{2\pi ik}}{1 - e^{2\pi ik/N}}. \end{aligned} \quad (8.32)$$

As long as $k \neq 0, N$ the numerator is 0. In the special cases that $k = 0, N$, we have $e^{2\pi ink/N} = 1$. So,

$$\sum_{n=0}^{N-1} e^{2\pi ink/N} = \sum_{n=0}^{N-1} 1 = N.$$

Therefore,

$$\sum_{n=0}^{N-1} \cos\left(\frac{2\pi nk}{N}\right) + i \sum_{n=0}^{N-1} \sin\left(\frac{2\pi nk}{N}\right) = \begin{cases} 0, & k = 1, \dots, N-1 \\ N, & k = 0, N \end{cases} \quad (8.33)$$

and the result follows.

We can use this to establish orthogonality relations of the following type for $p, q = 1, \dots, \frac{N}{2}$:

$$\sum_{n=0}^{N-1} \cos\left(\frac{2\pi pn}{N}\right) \cos\left(\frac{2\pi qn}{N}\right) = \frac{1}{2} \sum_{n=0}^{N-1} \left[\cos\left(\frac{2\pi(p-q)n}{N}\right) + \cos\left(\frac{2\pi(p+q)n}{N}\right) \right]. \quad (8.34)$$

Splitting the above sum into two sums and then evaluating the separate sums from earlier in this section,

$$\sum_{n=0}^{N-1} \cos\left(\frac{2\pi(p-q)n}{N}\right) = \begin{cases} 0, & p \neq q \\ N, & p = q \end{cases}$$

$$\sum_{n=0}^{N-1} \cos\left(\frac{2\pi(p+q)n}{N}\right) = \begin{cases} 0, & p+q \neq N \\ N, & p+q = N \end{cases}$$

we obtain

$$\sum_{n=0}^{N-1} \cos\left(\frac{2\pi pn}{N}\right) \cos\left(\frac{2\pi qn}{N}\right) = \begin{cases} N/2, & p = q \neq N/2 \\ N, & p = q = N/2 \\ 0, & \text{otherwise} \end{cases}. \quad (8.35)$$

Similarly, we find

$$\sum_{n=0}^{N-1} \sin\left(\frac{2\pi pn}{N}\right) \cos\left(\frac{2\pi qn}{N}\right) = \frac{1}{2} \sum_{n=0}^{N-1} \left[\sin\left(\frac{2\pi(p-q)n}{N}\right) + \sin\left(\frac{2\pi(p+q)n}{N}\right) \right] = 0. \quad (8.36)$$

and

$$\sum_{n=0}^{N-1} \sin\left(\frac{2\pi pn}{N}\right) \sin\left(\frac{2\pi qn}{N}\right) = \frac{1}{2} \sum_{n=0}^{N-1} \left[\cos\left(\frac{2\pi(p-q)n}{N}\right) - \cos\left(\frac{2\pi(p+q)n}{N}\right) \right] = \begin{cases} N/2, & p = q \neq N/2 \\ 0, & \text{otherwise} \end{cases}. \quad (8.37)$$

8.4.3 The Discrete Fourier Transform

The derivation of the coefficients for the DFT is now easily done using the discrete orthogonality from the last section. We start with the expansion

$$y_n = \frac{1}{2}a_0 + \sum_{p=1}^{N/2} \left[a_p \cos\left(\frac{2\pi pn}{N}\right) + b_p \sin\left(\frac{2\pi pn}{N}\right) \right], \quad n = 0, \dots, N-1. \quad (8.38)$$

We first sum over n :

$$\begin{aligned} \sum_{n=0}^{N-1} y_n &= \sum_{n=0}^{N-1} \left(\frac{1}{2}a_0 + \sum_{p=1}^{N/2} \left[a_p \cos\left(\frac{2\pi pn}{N}\right) + b_p \sin\left(\frac{2\pi pn}{N}\right) \right] \right) \\ &= \frac{1}{2}a_0 \sum_{n=0}^{N-1} 1 + \sum_{p=1}^{N/2} \left[a_p \sum_{n=0}^{N-1} \cos\left(\frac{2\pi pn}{N}\right) + b_p \sum_{n=0}^{N-1} \sin\left(\frac{2\pi pn}{N}\right) \right] \\ &= \frac{1}{2}a_0 N + \sum_{p=1}^{N/2} [a_p \cdot 0 + b_p \cdot 0] \\ &= \frac{1}{2}a_0 N. \end{aligned} \quad (8.39)$$

Therefore, we have $a_0 = \frac{2}{N} \sum_{n=0}^{N-1} y(t_n)$

Now, we can multiply both sides of the expansion by $\cos\left(\frac{2\pi qn}{N}\right)$ and sum over n .

$$\begin{aligned} \sum_{n=0}^{N-1} y_n \cos\left(\frac{2\pi qn}{N}\right) &= \sum_{n=0}^{N-1} \left(\frac{1}{2}a_0 + \sum_{p=1}^{N/2} \left[a_p \cos\left(\frac{2\pi pn}{N}\right) + b_p \sin\left(\frac{2\pi pn}{N}\right) \right] \right) \cos\left(\frac{2\pi qn}{N}\right) \\ &= \frac{1}{2}a_0 \sum_{n=0}^{N-1} \cos\left(\frac{2\pi qn}{N}\right) \\ &\quad + \sum_{p=1}^{N/2} \left[a_p \sum_{n=0}^{N-1} \cos\left(\frac{2\pi qn}{N}\right) + b_p \sum_{n=0}^{N-1} \sin\left(\frac{2\pi pn}{N}\right) \cos\left(\frac{2\pi qn}{N}\right) \right] \\ &= \begin{cases} \sum_{p=1}^{N/2} [a_p \frac{N}{2} \delta_{p,q} + b_p \cdot 0], & q \neq N/2, \\ \sum_{p=1}^{N/2} [a_p N \delta_{p,N/2} + b_p \cdot 0], & q = N/2, \end{cases} \\ &= \begin{cases} \frac{1}{2}a_q N, & q \neq N/2 \\ a_{N/2} N, & q = N/2. \end{cases} \end{aligned} \quad (8.40)$$

So, we have found that

$$a_q = \frac{2}{N} \sum_{n=0}^{N-1} y(t_n) \cos\left(\frac{2\pi qn}{N}\right), \quad q \neq \frac{N}{2}, \quad (8.41)$$

$$\begin{aligned}
a_{N/2} &= \frac{1}{N} \sum_{n=0}^{N-1} y(t_n) \cos\left(\frac{2\pi n(N/2)}{N}\right) \\
&= \frac{1}{N} \sum_{n=0}^{N-1} y(t_n) \cos(\pi n). \tag{8.42}
\end{aligned}$$

$$\begin{aligned}
&\text{Similarly, } \sum_{n=0}^{N-1} y_n \sin\left(\frac{2\pi qn}{N}\right) = \\
&= \sum_{n=0}^{N-1} \left(\frac{1}{2}a_0 + \sum_{p=1}^{N/2} \left[a_p \cos\left(\frac{2\pi pn}{N}\right) + b_p \sin\left(\frac{2\pi pn}{N}\right) \right] \right) \sin\left(\frac{2\pi qn}{N}\right) \\
&= \frac{1}{2}a_0 \sum_{n=0}^{N-1} \sin\left(\frac{2\pi qn}{N}\right) \\
&\quad + \sum_{p=1}^{N/2} \left[a_p \sum_{n=0}^{N-1} \cos\left(\frac{2\pi pn}{N}\right) \sin\left(\frac{2\pi qn}{N}\right) + b_p \sum_{n=0}^{N-1} \sin\left(\frac{2\pi pn}{N}\right) \sin\left(\frac{2\pi qn}{N}\right) \right] \\
&= \sum_{p=1}^{N/2} \left[a_p \cdot 0 + b_p \frac{N}{2} \delta_{p,q} \right] \\
&= \frac{1}{2}b_q N. \tag{8.43}
\end{aligned}$$

Finally, we have

$$b_q = \frac{2}{N} \sum_{n=0}^{N-1} y(t_n) \sin\left(\frac{2\pi qn}{N}\right), \quad q = 1, \dots, \frac{N}{2} - 1. \tag{8.44}$$

8.4.4 Applications

In the last section we saw that given a set of data, $y_n, n = 0, 1, \dots, N-1$, that one can construct the corresponding discrete, finite Fourier series. The series is given by

$$y_n = \frac{1}{2}a_0 + \sum_{p=1}^{N/2} \left[a_p \cos\left(\frac{2\pi pn}{N}\right) + b_p \sin\left(\frac{2\pi pn}{N}\right) \right], \quad n = 0, \dots, N-1. \tag{8.45}$$

and the Fourier coefficients were found as

$$\begin{aligned}
a_p &= \frac{2}{N} \sum_{n=1}^N y(t_n) \cos\left(\frac{2\pi pn}{N}\right), \quad p = 1, \dots, N/2 - 1 \\
b_p &= \frac{2}{N} \sum_{n=1}^N y(t_n) \sin\left(\frac{2\pi pn}{N}\right), \quad p = 1, 2, \dots, N/2 - 1 \\
a_0 &= \frac{1}{N} \sum_{n=1}^N y(t_n), \\
a_{N/2} &= \frac{1}{N} \sum_{n=1}^N y(t_n) \cos n\pi, \\
b_0 &= b_{N/2} = 0
\end{aligned} \tag{8.46}$$

In this section we show how this is implemented using MATLAB. Consider the data¹ of monthly mean surface temperatures at Amthritrite Point, Canada shown in Table 8.1. The temperature was recorded in °C and averaged for each month over a two year period. We would like to look for the frequency content of this time series.

Month	1	2	3	4	5	6	7	8	9	10	11	12
1982	7.6	7.4	8.2	9.2	10.2	11.5	12.4	13.4	13.7	11.8	10.1	9.0
1983	8.9	9.5	10.6	11.4	12.9	12.7	13.9	14.2	13.5	11.4	10.9	8.1

Table 8.1. Monthly mean surface temperatures (°C) at Amthritrite Point, Canada for 1982-1983.

In Figure 8.8 we plot the above data as circles. We then use the data to compute the Fourier coefficients. These coefficients are used in the discrete Fourier series an plotted on top of the data in red. We see that the reconstruction fits the data.

The implementations of DFT are done using MATLAB. We provide the code at the end of this section.

Generally, we are interested in determining the frequency content of a signal. For example, we consider a pure note,

$$y(t) = \sin(10\pi t).$$

Sampling this signal with $N = 128$ points on the interval $[0, 5]$, we find the discrete Fourier coefficients as shown in Figure 8.9. Note the spike at the right place in the B plot. The others spikes are actually very small if you look at the scale on the plot of A coefficients.

One can use these coefficients to reconstruct the signal. This is shown in Figure 8.10

We can look at more interesting functions. For example, what if we add two pure notes together, such as

¹ This example is from *Data Analysis Methods in Physical Oceanography*, W. J. Emery and R.E. Thomson, Elsevier, 1997.

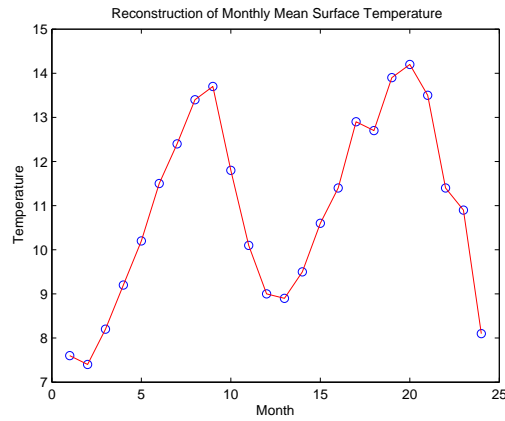


Fig. 8.8. Plot and reconstruction of the monthly mean surface temperature data.

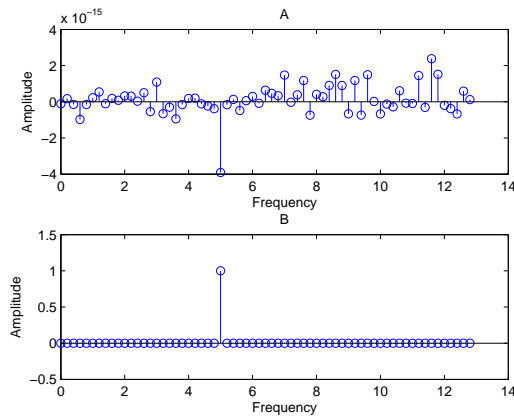


Fig. 8.9. Computed discrete Fourier coefficients for $y(t) = \sin(10\pi t)$, with $N = 128$ points on the interval $[0, 5]$.

$$y(t) = \sin(10\pi t) - \frac{1}{2} \cos(6\pi t).$$

We see from Figure 8.11 that the implementation works. The Fourier coefficients for a slightly more complicated signal,

$$y(t) = e^{\alpha t} \sin(10\pi t)$$

for $\alpha = 0.1$, is shown in Figure 8.12 and the corresponding reconstruction is shown in Figure 8.13. We will look into more interesting features in discrete signals later in the chapter.

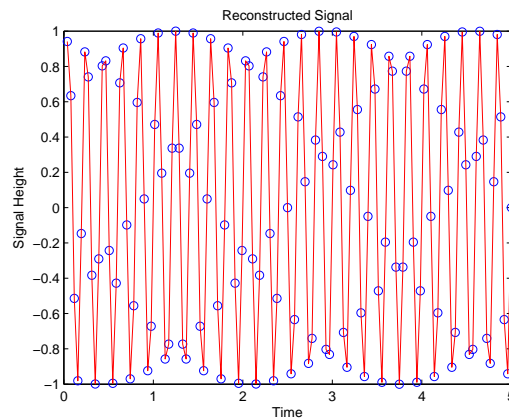


Fig. 8.10. Reconstruction of $y(t) = \sin(10\pi t)$ from its Fourier coefficients.

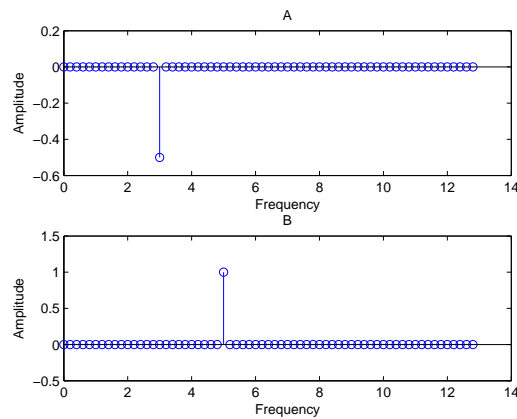


Fig. 8.11. Computed discrete Fourier coefficients for $\sin(10\pi t) - \frac{1}{2} \cos(6\pi t)$ with $N = 128$ points on the interval $[0, 5]$.

8.4.5 MATLAB Implementation

In this section we provide implementations of the discrete trigonometric transform in MATLAB. The first implementation is a straightforward one which can be done in most programming languages. The second implementation makes use of matrix computations that can be performed in MATLAB. Sums can be done with matrix multiplication, as describes in Appendix I. This eliminates the loops in the first program below and speeds up the computation for large data sets.

Direct Implementation

The following code was used to produce Figure 8.8.

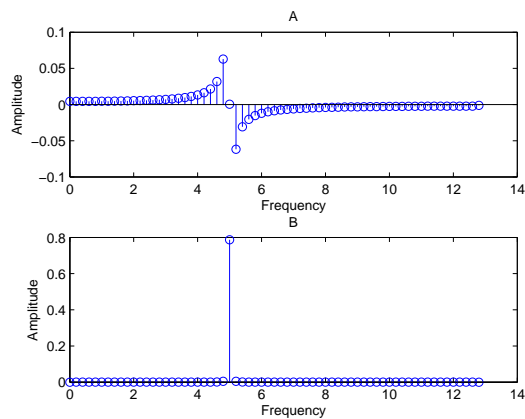


Fig. 8.12. Computed discrete Fourier coefficients for $y(t) = e^{\alpha t} \sin(10\pi t)$ with $\alpha = 0.1$ and $N = 128$ points on the interval $[0, 5]$.

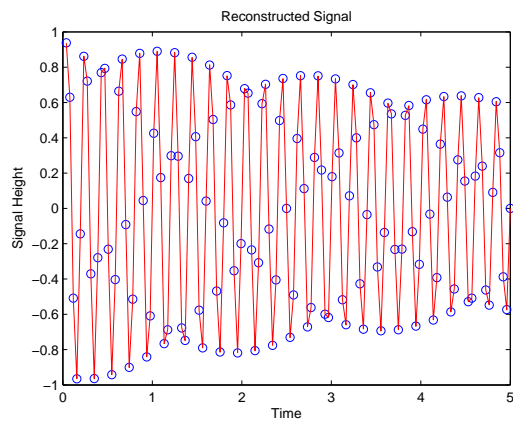


Fig. 8.13. Reconstruction of $y(t) = e^{\alpha t} \sin(10\pi t)$ with $\alpha = 0.1$ from its Fourier coefficients.

```
%
% DFT in a direct implementation
%
% Enter Data in y
y=[7.6 7.4 8.2 9.2 10.2 11.5 12.4 13.4 13.7 11.8 10.1 ...
   9.0 8.9 9.5 10.6 11.4 12.9 12.7 13.9 14.2 13.5 11.4 10.9 8.1];
% Get length of data vector or number of samples
N=length(y);
% Compute Fourier Coefficients
for p=1:N/2+1
```

```

    A(p)=0;
    B(p)=0;
    for n=1:N
        A(p)=A(p)+2/N*y(n)*cos(2*pi*(p-1)*n/N)';
        B(p)=B(p)+2/N*y(n)*sin(2*pi*(p-1)*n/N)';
    end
end
A(N/2+1)=A(N/2+1)/2;
% Reconstruct Signal - pmax is number of frequencies used in increasing order
pmax=13; for n=1:N
    ynew(n)=A(1)/2;
    for p=2:pmax
        ynew(n)=ynew(n)+A(p)*cos(2*pi*(p-1)*n/N)+B(p)*sin(2*pi*(p-1)*n/N);
    end
end
end
% Plot Data
plot(y,'o')
% Plot reconstruction over data
hold on
plot(ynew,'r')
hold off
title('Reconstruction of Monthly
Mean Surface Temperature')
xlabel('Month')
ylabel('Temperature')

```

The next routine shows how we can determine the spectral content of a signal, given in this case by a function and not a measured time series.

```

% IMPLEMENTATION OF DFT USING TRIGONOMETRIC FORM
% N = Number of samples
% T = Record length in time
% y = Sampled signal
%
N=128; T=5; dt=T/N; t=(1:N)*dt; f0=5; alpha=0.1;
y=exp(-alpha*t).*sin(2*pi*f0*t);

% Compute arguments of trigonometric functions
for n=1:N
    for p=0:N/2
        Phi(p+1,n)=2*pi*p*n/N;
    end
end

% Compute Fourier Coefficients
for p=1:N/2+1

```

```

    A(p)=2/N*y*cos(Phi(p,:))';
    B(p)=2/N*y*sin(Phi(p,:))';
end
A(1)=2/N*sum(y); A(N/2+1)=A(N/2+1)/2; B(N/2+1)=0;

% Reconstruct Signal - pmax is number of frequencies used in increasing order
pmax=N/2;
for n=1:N
    ynew(n)=A(1)/2;
    for p=2:pmax
        ynew(n)=ynew(n)+A(p)*cos(Phi(p,n))+B(p)*sin(Phi(p,n));
    end
end

% Plot Data
figure(1)
plot(t,y,'o')

% Plot reconstruction over data
hold on
plot(t,ynew,'r')
xlabel('Time')
ylabel('Signal Height')
title('Reconstructed Signal')
hold off

% Compute Frequencies
n2=N/2; f=(0:n2)/(n2*2*dt);

% Plot Fourier Coefficients
figure(2)
subplot(2,1,1) s
stem(f,A)
xlabel('Frequency')
ylabel('Amplitude')
title('A')
subplot(2,1,2)
stem(f,B)
xlabel('Frequency')
ylabel('Amplitude')
title('B')

% Plot Fourier Spectrum
figure(3)
Power=sqrt(A.^2+B.^2);

```

```

stem(f,Power(1:n2+1))
xlabel('Frequency')
ylabel('Power')
title('Periodogram')

```

Compact Implementation

This implementation uses matrix products and is described in Appendix G.

```

%
% DFT in a compact implementation
%
% Enter Data in y
y=[7.6 7.4 8.2 9.2 10.2 11.5 12.4 13.4 13.7 11.8 10.1 ...
    9.0 8.9 9.5 10.6 11.4 12.9 12.7 13.9 14.2 13.5 11.4 10.9 8.1];
N=length(y);

% Compute the matrices of trigonometric functions
p=1:N/2+1;
n=1:N;
C=cos(2*pi*n'*(p-1)/N);
S=sin(2*pi*n'*(p-1)/N);

% Compute Fourier Coefficients
A=2/N*y*C;
B=2/N*y*S;
A(N/2+1)=A(N/2+1)/2;

% Reconstruct Signal - pmax is number of frequencies used in increasing order
pmax=13;
ynew=A(1)/2+C(:,2:pmax)*A(2:pmax)'+S(:,2:pmax)*B(2:pmax)';

% Plot Data
plot(y,'o')

% Plot reconstruction over data
hold on
plot(ynew,'r')
hold off
title('Reconstruction of Monthly
Mean Surface Temperature')
xlabel('Month')
ylabel('Temperature')

```

8.4.6 The Discrete Exponential Transform

The derivation of the coefficients for the DFT was obtained using the discrete orthogonality in the last section. However, this is not the form used in MATLAB for spectral analysis. MATLAB allows for the computation of the Fast Fourier Transform (FFT) and its description in the help section does not involve sines and cosines. Namely, MATLAB defines the transform and inverse transform as

For length N input vector x , the DFT is a length N vector X ,

with elements

$$X(k) = \sum_{n=1}^N x(n) \exp(-j*2*\pi*(k-1)*(n-1)/N), \quad 1 \leq k \leq N.$$

The inverse DFT (computed by IFFT) is given by

$$x(n) = (1/N) \sum_{k=1}^N X(k) \exp(j*2*\pi*(k-1)*(n-1)/N), \quad 1 \leq n \leq N.$$

Or, it also provides the following:

$$F_k = \sum_{j=0}^{N-1} W_N^{jk} f_j. \quad (8.47)$$

where $W_N = e^{-2\pi i/N}$.

In this section we will derive the discrete Fourier exponential transform and find the relationship between what MATLAB is computing and the discrete Fourier trigonometric series in the past sections. This will also be useful as a preparation for a discussion of FFT in the next section.

We will start with the DFT:

$$y(t_n) = \frac{1}{2}a_0 + \sum_{p=1}^{N/2} \left[a_p \cos\left(\frac{2\pi pn}{N}\right) + b_p \sin\left(\frac{2\pi pn}{N}\right) \right]. \quad (8.48)$$

We use Euler's formula to rewrite the trigonometric functions in terms of exponentials. The DFT formula can be written as

$$\begin{aligned} y(t_n) &= \frac{1}{2}a_0 + \sum_{p=1}^{N/2} \left[a_p \frac{e^{2\pi ipn/N} + e^{-2\pi ipn/N}}{2} + b_p \frac{e^{2\pi ipn/N} - e^{-2\pi ipn/N}}{2i} \right] \\ &= \frac{1}{2}a_0 + \sum_{p=1}^{N/2} \left[\frac{1}{2} (a_p - ib_p) e^{2\pi ipn/N} + \frac{1}{2} (a_p + ib_p) e^{-2\pi ipn/N} \right]. \end{aligned} \quad (8.49)$$

We define $C_p = \frac{1}{2} (a_p - ib_p)$ and note that the above result can be written as

$$y(t_n) = C_0 + \sum_{p=1}^{N/2} \left[C_p e^{2\pi ipn/N} + \bar{C}_p e^{-2\pi ipn/N} \right]. \quad (8.50)$$

The terms in the sums look similar. We can actually combine them into one form. Note that $e^{2\pi iN} = \cos(2\pi N) + i \sin(2\pi N) = 1$. Thus, we can write

$$e^{-2\pi ipn/N} = e^{-2\pi ipn/N} e^{-2\pi iN} = e^{2\pi i(N-p)n/N}$$

in the second sum. Since $p = 1, \dots, N/2$, we see that $N - p = N - 1, N - 2, \dots, N/2$. So, we can rewrite the second sum as

$$\sum_{p=1}^{N/2} \bar{C}_p e^{-2\pi ipn/N} = \sum_{p=1}^{N/2} \bar{C}_p e^{2\pi i(N-p)n/N} = \sum_{q=N/2}^{N-1} \bar{C}_{N-q} e^{2\pi iqn/N}.$$

Since q is a *dummy* index (it can be replaced by any letter without changing the value of the sum), we can replace it with a p and combine the terms in both sums to obtain

$$y(t_n) = \sum_{p=0}^{N-1} Y_p e^{2\pi ipn/N}, \quad (8.51)$$

where

$$Y_p = \begin{cases} \frac{a_0}{2}, & p = 0, \\ \frac{1}{2}(a_p - ib_p), & 0 < p < N/2, \\ a_{N/2}, & p = N/2, \\ \frac{1}{2}(a_{N-p} + ib_{N-p}), & N/2 < p < N. \end{cases} \quad (8.52)$$

Notice that the real and imaginary parts of the Fourier coefficients obey certain symmetry properties over the full range of the indices since the real and imaginary parts are related between $p \in [0, N/2]$ and $p \in [N/2, N - 1]$. [A description of this will be provided later.]

We can now determine the coefficients in terms of the sampled data.

$$\begin{aligned} C_p &= \frac{1}{2}(a_p - ib_p) \\ &= \frac{1}{N} \sum_{n=1}^N y(t_n) \left[\cos\left(\frac{2\pi pn}{N}\right) - i \sin\left(\frac{2\pi pn}{N}\right) \right] \\ &= \frac{1}{N} \sum_{n=1}^N y(t_n) e^{-2\pi ipn/N}. \end{aligned} \quad (8.53)$$

Thus,

$$Y_p = \frac{1}{N} \sum_{n=1}^N y(t_n) e^{-2\pi ipn/N}, \quad 0 < p < \frac{N}{2} \quad (8.54)$$

and

$$\begin{aligned}
Y_p &= \bar{C}_{N-p} \\
&= \frac{1}{N} \sum_{n=1}^N y(t_n) e^{2\pi i(N-p)n/N}, \quad \frac{N}{2} < p < N \\
&= \frac{1}{N} \sum_{n=1}^N y(t_n) e^{-2\pi i p n/N}.
\end{aligned} \tag{8.55}$$

We have shown that for all Y 's but two, the form is

$$Y_p = \frac{1}{N} \sum_{n=1}^N y(t_n) e^{-2\pi i p n/N}. \tag{8.56}$$

However, we can easily show that this is also true when $p = 0$ and $p = \frac{N}{2}$.

$$\begin{aligned}
Y_{N/2} &= a_{N/2} \\
&= \frac{1}{N} \sum_{n=1}^N y(t_n) \cos n\pi \\
&= \frac{1}{N} \sum_{n=1}^N y(t_n) [\cos n\pi - i \sin n\pi] \\
&= \frac{1}{N} \sum_{n=1}^N y(t_n) e^{-2\pi i n(N/2)/N}
\end{aligned} \tag{8.57}$$

and

$$\begin{aligned}
Y_0 &= \frac{1}{2} a_0 \\
&= \frac{1}{N} \sum_{n=1}^N y(t_n) \\
&= \frac{1}{N} \sum_{n=1}^N y(t_n) e^{2\pi i n(0)/N}
\end{aligned} \tag{8.58}$$

Thus, all of the F_p 's are of the same form. This gives us the discrete transform pair

$$y(t_n) = \sum_{p=0}^{N-1} Y_p e^{2\pi i p n/N}, \tag{8.59}$$

$$Y_p = \frac{1}{N} \sum_{n=1}^N y(t_n) e^{-2\pi i p n/N}. \tag{8.60}$$

Note that this is similar to the definition of the FFT given in MATLAB.

8.4.7 FFT: The Fast Fourier Transform

The usual computation of the discrete Fourier transform is done using the Fast Fourier Transform (FFT). There are various implementations of it, but a standard form is the Radix-2 FFT. We describe this FFT in the current section. We begin by writing the DFT compactly using $W = e^{-2\pi i/N}$. Note that $W^{N/2} = -1$, $W^N = 1$, and $e^{2\pi ijk/N} = W^{jk}$. We can then write

$$F_k = \sum_{j=0}^{N-1} W^{jk} f_j. \quad (8.61)$$

The key to the FFT is that this sum can be written as two similar sums:

$$\begin{aligned} F_k &= \sum_{j=0}^{N-1} W^{jk} f_j \\ &= \sum_{j=0}^{N/2-1} W^{jk} f_j + \sum_{j=N/2}^{N-1} W^{jk} f_j \\ &= \sum_{j=0}^{N/2-1} W^{jk} f_j + \sum_{m=0}^{N/2-1} W^{k(m+N/2)} f_{m+N/2}, \quad \text{for } m = j - \frac{N}{2} \\ &= \sum_{j=0}^{N/2-1} \left[W^{jk} f_j + W^{k(j+N/2)} f_{j+N/2} \right] \\ &= \sum_{j=0}^{N/2-1} W^{jk} [f_j + (-1)^k f_{j+N/2}] \end{aligned} \quad (8.62)$$

since $W^{k(j+N/2)} = W^{kj}(W^{N/2})^k$ and $W^{N/2} = -1$.

Thus, the sum appears to be of the same form as the initial sum, but there are half as many terms with a different coefficient for the W^{jk} 's. In fact, we can separate the terms involving the + or - sign by looking at the even and odd values of k .

For even $k = 2m$, we have

$$F_{2m} = \sum_{j=0}^{N/2-1} (W^{2m})^j [f_j + f_{j+N/2}], \quad m = 0, \dots, \frac{N}{2} - 1. \quad (8.63)$$

For odd $k = 2m + 1$, we have

$$F_{2m+1} = \sum_{j=0}^{N/2-1} (W^{2m})^j W^j [f_j - f_{j+N/2}], \quad m = 0, \dots, \frac{N}{2} - 1. \quad (8.64)$$

Each of these equations gives the Fourier coefficients in terms of a similar sum using fewer terms and with a different weight, $W^2 = (e^{-2\pi i/N})^2 =$

$e^{-2\pi i/(N/2)}$. If N is a power of 2, then this process can be repeated over and over until one ends up with a simple sum.

The process is easily seen when written out for a small number of samples. Let $N = 8$. Then a first pass at the above gives

$$\begin{aligned}
F_0 &= f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 \\
F_1 &= f_0 + Wf_1 + W^2f_2 + W^3f_3 + W^4f_4 + W^5f_5 + W^6f_6 + W^7f_7 \\
F_2 &= f_0 + W^2f_1 + W^4f_2 + W^6f_3 + f_4 + W^2f_5 + W^4f_6 + W^6f_7 \\
F_3 &= f_0 + W^3f_1 + W^6f_2 + Wf_3 + W^4f_4 + W^7f_5 + W^2f_6 + W^5f_7 \\
F_4 &= f_0 + W^4f_1 + f_2 + W^4f_3 + f_4 + W^4f_5 + f_6 + W^4f_7 \\
F_5 &= f_0 + W^5f_1 + W^2f_2 + W^7f_3 + W^4f_4 + Wf_5 + W^6f_6 + W^3f_7 \\
F_6 &= f_0 + W^6f_1 + W^4f_2 + W^2f_3 + f_4 + W^6f_5 + W^4f_6 + W^2f_7 \\
F_7 &= f_0 + W^7f_1 + W^6f_2 + W^5f_3 + W^4f_4 + W^3f_5 + W^2f_6 + Wf_7
\end{aligned} \tag{8.65}$$

The point is that the terms in these expressions can be regrouped with $W = e^{-\pi i/8}$ and noting $W^4 = -1$:

$$\begin{aligned}
F_0 &= (f_0 + f_4) + (f_1 + f_5) + (f_2 + f_6) + (f_3 + f_7) \\
&\equiv g_0 + g_1 + g_2 + g_3 \\
F_1 &= (f_0 - f_4) + (f_1 - f_5)W + (f_2 - f_6)W^2 + (f_3 - f_7)W^3 \\
&\equiv g_4 + g_6 + g_5 + g_7 \\
F_2 &= (f_0 + f_4) + (f_1 + f_5)W^2 - (f_2 + f_6) - (f_3 + f_7)W^2 \\
&= g_0 - g_2 + (g_1 - g_3)W^2 \\
F_3 &= (f_0 - f_4) - (f_2 - f_6)W + (f_1 - f_5)WW^2 + (f_3 - f_7)WW^6 \\
&= g_4 - g_6 + g_5W^2 + g_7W^6 \\
F_4 &= (f_0 + f_4) + (f_1 + f_5) - (f_2 + f_6) - (f_3 + f_7) \\
&= g_0 + g_2 - g_1 - g_3 \\
F_5 &= (f_0 - f_4) + (f_2 - f_6)W + (f_1 - f_5)WW^4 + (f_3 - f_7)WW^4 \\
&= g_4 + g_6 + g_5W^4 + g_7W^4 \\
F_6 &= (f_0 + f_4) + (f_1 + f_5)W^6 - (f_2 + f_6) - (f_3 + f_7)W^6 \\
&= g_0 - g_2 + (g_1 - g_3)W^6 \\
F_7 &= (f_0 - f_4) - (f_2 - f_6)W + (f_1 - f_5)WW^6 + (f_3 - f_7)WW^2 \\
&= g_4 - g_6 + g_5W^6 + g_7W^2
\end{aligned} \tag{8.66}$$

However, each of the g -series can be rewritten as well, leading to

$$\begin{aligned}
F_0 &= (g_0 + g_2) + (g_1 + g_3) \equiv h_0 + h_1 \\
F_1 &= (g_4 + g_6) + (g_5 + g_7) \equiv h_4 + h_5 \\
F_2 &= (g_0 - g_2) + (g_1 - g_3)W^2 \equiv h_2 + h_3
\end{aligned}$$

$$\begin{aligned}
 F_3 &= (g_4 - g_6) + (g_5 - g_7)W^2 \equiv h_6 + h_7 \\
 F_4 &= (g_0 + g_2) - (g_1 + g_3) = h_0 - h_1 \\
 F_5 &= (g_4 + g_6) - (g_5 + g_7) = h_4 - h_5 \\
 F_6 &= g_0 - g_2 - (g_1 - g_3)W^2 = h_2 - h_3 \\
 F_7 &= g_4 - g_6 + g_5W^6 + g_7W^2 = h_6 - h_7
 \end{aligned}
 \tag{8.67}$$

Thus, the computation of the Fourier coefficients amounts to inputting the f 's and computing the g 's. This takes 8 additions and 4 multiplications. Then one get the h 's, which is another 8 additions and 4 multiplications. There are three stages, amounting to a total of 12 multiplications and 24 additions. Carrying out the process in general, one has $\log_2 N$ steps with $N/2$ multiplications and N additions per step. In the direct computation one has $(N - 1)^2$ multiplications and $N(N - 1)$ additions. Thus, for $N = 8$, that would be 49 multiplications and 56 additions.

The above process is typically shown schematically in a "butterfly diagram". The basic butterfly transformation is displayed in Figure 8.14. An 8 point FFT is shown in Figure 8.15.

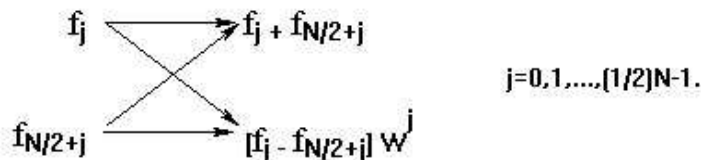


Fig. 8.14. This is the basic FFT butterfly.

In the actual implementation, one computes with the h 's in the following order:

Output and Binary Representation	Desired Order
$h_0 + h_1 = F_0, \quad 000$	$F_0, \quad 000$
$h_0 - h_1 = F_4, \quad 100$	$F_1, \quad 001$
$h_2 + h_3 = F_2, \quad 010$	$F_2, \quad 010$
$h_2 - h_3 = F_6, \quad 110$	$F_3, \quad 011$
$h_4 + h_5 = F_1, \quad 001$	$F_4, \quad 100$
$h_4 - h_5 = F_5, \quad 101$	$F_5, \quad 101$
$h_6 + h_7 = F_3, \quad 011$	$F_6, \quad 110$
$h_6 - h_7 = F_7, \quad 111$	$F_7, \quad 111$

The binary representation of the index was also listed. Notice that the output is in bit-reversed order as compared to the right side of the table which shows the coefficients in the correct order. [Just compare the columns

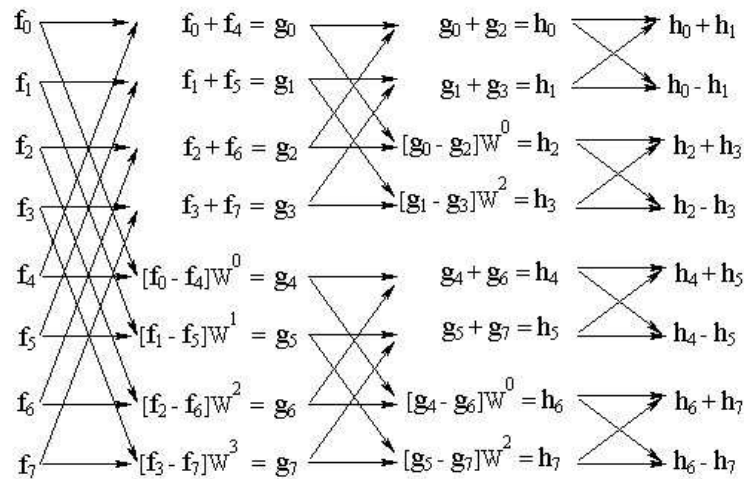


Fig. 8.15. This is an 8 point FFT butterfly diagram.

in each set of binary representations.] So, typically there is a bit reversal routine needed to unscramble the order of the output coefficients in order to use them.

8.5 Appendix

8.5.1 Matrix Operations for MATLAB

The beauty of using MATLAB is that many operations can be performed using matrix operations and that one can perform complex arithmetic. This eliminates many loops and make the coding of computations quicker. However, one needs to be able to understand the formalism. In this section we elaborate on these operations so that one can see how the MATLAB implementation of the direct computation of the DFT can be carried out in compact form as shown previously in the MATLAB Implementation section. This is all based upon the structure of MATLAB, which is essentially a MATrix LABoratory.

A key operation between matrices is matrix multiplication. An $n \times m$ matrix is simply a collection of numbers arranged in n rows and m columns. For example, the matrix $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ is a 2×3 matrix. The entries (elements) of a general matrix A can be represented as a_{ij} which represents the i th row and j th column.

Given two matrices, A and B , we can define the multiplication of these matrices when the number of columns of A equals the number of rows of B . The product, which we represent as matrix C , is given by the ij th element of

C . In particular, we let A be a $p \times m$ matrix and B an $m \times q$ matrix. The product, C , will be a $p \times q$ matrix with entries

$$\begin{aligned} C_{ij} &= \sum_{k=1}^m a_{ik}b_{kj}, \quad i = 1, \dots, p, \quad j = 1, \dots, q, \\ &= a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{im}b_{mj}. \end{aligned} \quad (8.68)$$

If we wanted to compute the sum $\sum_{n=1}^N a_n b_n$, then in a typical programming language we could use a loop, such as

```
Sum =0
Loop n from 1 to N
    Sum = Sum + a(n)*b(n)
End Loop
```

In MATLAB we could do this with a loop as above, or we could resort to matrix multiplication. We can let a and b be $1 \times n$ and $n \times 1$ matrices, respectively. Then the product would be a 1×1 matrix; namely, the sum we are seeking. However, these matrices are not always of the suggested size.

A $1 \times n$ matrix is called a row vector and a $n \times 1$ matrix is called a column vector. Often we have that both are of the same type. One can convert a row vector into a column vector, or vice versa, using the matrix operation called a *transpose*. More generally, the transpose of a matrix is defined as follows: A^T has the elements satisfying $(A^T)_{ij} = a_{ji}$. In MATLAB, the transpose of a matrix A is A' .

Thus, if we want to perform the above sum, we have $\sum_{n=1}^N a_n b_n = \sum_{n=1}^N a_{1n} b_{n1}$. In particular, if both \mathbf{a} and \mathbf{b} are row vectors, the sum in MATLAB is given by $\mathbf{a}\mathbf{b}'$, and if they are both column vectors, the sum is $\mathbf{a}'\mathbf{b}$. This notation is much easier to type.

In our computation of the DFT, we have many sums. For example, we want to compute the coefficients of the sine functions,

$$b_p = \frac{2}{N} \sum_{n=1}^N y(t_n) \sin\left(\frac{2\pi p n}{N}\right), \quad p = 0, \dots, N/2 \quad (8.69)$$

The sum can be computed as a matrix product. The function y only has values at times t_n . This is the sampled data. We can represent it as a vector. The sine functions take values at arguments (angles) depending upon p and n . So, we can represent the sines as an $N \times (N/2 + 1)$ or $(N/2 + 1) \times N$ matrix. The Fourier coefficient thus becomes a simple matrix multiplication, ignoring the prefactor $\frac{2}{N}$. Thus, if we put the sampled data in a $1 \times N$ vector \mathbf{Y} and put the sines in an $N \times (\frac{N}{2} + 1)$ vector \mathbf{S} , the Fourier coefficient will be the product, which has size $1 \times (\frac{N}{2} + 1)$. Thus, in the code we see that these coefficients are computed as $\mathbf{B} = \mathbf{2}/\mathbf{N} * \mathbf{y} * \mathbf{S}$ for the given \mathbf{y} and \mathbf{B} matrices. The A coefficients are computed in the same manner. Comparing

the two codes in that section, we see how much easier it is to implement. However, the number of multiplications and additions has not decreased. This is why the FFT is generally better. But, seeing the direct implementation helps one to understand what is being computed before seeking a more efficient implementation, such as the FFT.